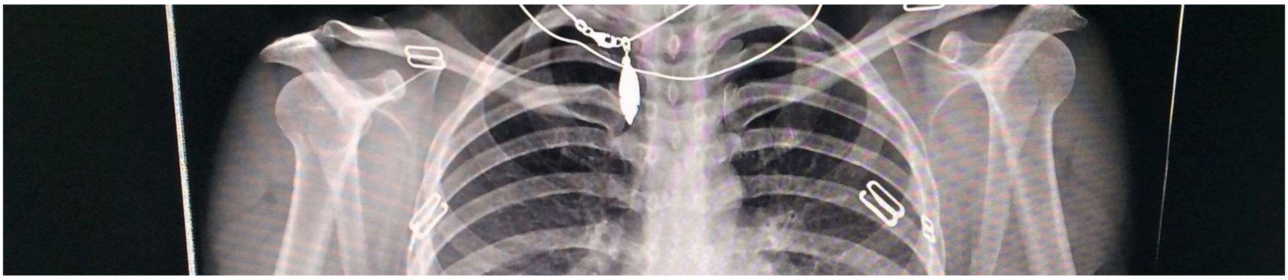
Chest X-Ray Images (Pneumonia)

- Mayank Srivastava

LINKEDIN CONNECT

- (https://www.linkedin.com/in/mayank-srivastava-6a8421105/)



Dataset: https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia (https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia)

Content

- The dataset is organized into 3 folders (train, test, val) and contains subfolders for each image category (Pneumonia/Normal).
- There are 5,863 X-Ray images (JPEG) and 2 categories (Pneumonia/Normal).

Source

- Chest X-ray images (anterior-posterior) were selected from retrospective cohorts of pediatric patients of one to five years old from Guangzhou Women and Children's Medical Center, Guangzhou.
- All chest X-ray imaging was performed as part of patients' routine clinical care.
- For the analysis of chest x-ray images, all chest radiographs were initially screened for quality control by removing all low quality or unreadable scans.
- The diagnoses for the images were then graded by two expert physicians before being cleared for training the AI system. - - In order to account for any grading errors, the evaluation set was also checked by a third expert.

Project Overview

- This is a Classification Problem.
- **Objective:** To classify chest X-ray images into two categories: Pneumonia and Normal using a CNN model.

Importing Libraries

In [1]:

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam  # You can try different optimizers
```

In [2]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)

from sklearn.metrics import recall_score,confusion_matrix,roc_auc_score, roc_curve,auc
```

Setting the direcory for train, test, val image data sets

In [3]:

```
train_dir = r'E:\DS journey\Deep Learning Datasets\chest_xray\train'
test_dir = r'E:\DS journey\Deep Learning Datasets\chest_xray\test'
val_dir = r'E:\DS journey\Deep Learning Datasets\chest_xray\val'
```

In [97]:

```
# Data Distribution

data=pd.DataFrame({
    "Train":{i:len(os.listdir(os.path.join(train_dir,i))) for i in os.listdir(train_dir)},
    "Test":{i:len(os.listdir(os.path.join(test_dir,i))) for i in os.listdir(test_dir)},
                  })
data
```

Out[97]:

	Train	Test
NORMAL	1341	234
PNEUMONIA	3875	390

In [95]:

```
list(data.index)
```

Out[95]: ['NORMAL', 'PNEUMONIA']

In [98]:

```
[sum(data[i]) for i in data ]
```

Out[98]: [5216, 624]

In [114]:

```
sum(data.loc['NORMAL',:])
```

Out[114]: 1575

In [115]:

```
sum(data.loc['PNEUMONIA',:])
```

Out[115]: 4265

Data Distribution

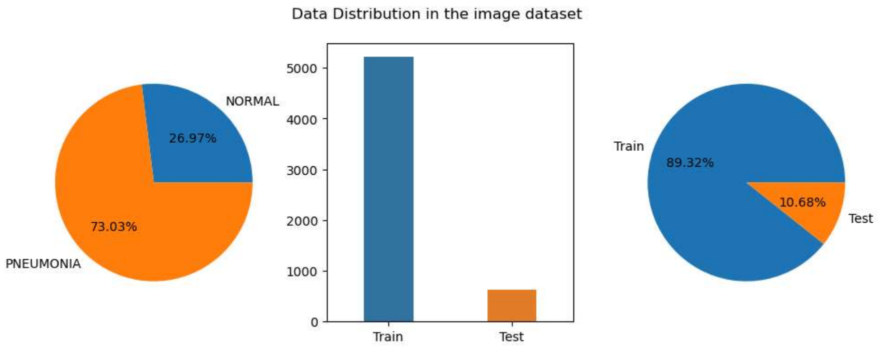
In [123]:

```
plt.figure(figsize =(12,4))
plt.subplot(1,3,1)
plt.pie([sum(data.loc[i,:]) for i in list(data.index)], labels =data.index, autopct ="%1.2f%%")


plt.subplot(1,3,2)
sns.barplot(x= data.columns, y=sum(data.values), width =0.4)

plt.subplot(1,3,3)
plt.pie([sum(data[i]) for i in list(data.columns)], labels =data.columns, autopct ="%1.2f%%")

plt.suptitle('Data Distribution in the image dataset')
plt.show()
```



Setting seed for Reproducibility

In [4]:

```
# Set random seeds for reproducibility
import random
seed = 42
tf.random.set_seed(seed)    # Sets the random seed for TensorFlow operations.
np.random.seed(seed)        # Sets the random seed for NumPy operations.
random.seed(seed)           # Sets the random seed for Python's built-in random module.

```

In [26]:

```
img_width, img_height =  240, 240  # Target image size
batch_size = 32
```

In [6]:

```
# Data generators with augmentation


# rotation_range=40,
# width_shift_range=0.2,
# height_shift_range=0.2,
# shear_range=0.2,
# zoom_range=0.2,
# horizontal_flip=True
```

Creating the ImageDataGenerator instance for test and train

In [7]:

```
train_datagen = ImageDataGenerator(rescale=1./255,
                                    rotation_range=40,
                                    width_shift_range=0.2,
                                    height_shift_range=0.2,
                                    shear_range=0.2,
                                    zoom_range=0.2,
                                    horizontal_flip=True)
test_datagen = ImageDataGenerator(rescale=1./255)
val_datagen = ImageDataGenerator(rescale=1./255)
```

In [8]:

```
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(240, 240),
    batch_size=32,
    class_mode='binary')  # Binary classification: cat or dog
```

```
Found 5216 images belonging to 2 classes.
```

Loading the image data from local direcotry

In [9]:

```
test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(240,240),
    batch_size=32,
    class_mode='binary')  # Binary classification: cat or dog
```

```
Found 624 images belonging to 2 classes.
```

In [10]:

```
val_generator = val_datagen.flow_from_directory(
    val_dir,
    target_size=(240,240),
    batch_size=batch_size,
    class_mode='binary')  # Binary classification: cat or dog
```

```
Found 16 images belonging to 2 classes.
```

CNN Model Architecture

In [11]:
```python
model = Sequential()

model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(240, 240, 3)))
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))

model.add(Flatten())

## ANN
model.add(Dense(128, activation='relu'))

model.add(Dense(64, activation='relu')) # hidden

model.add(Dropout(0.5))  # to prevent ovefitting
model.add(Dense(1, activation='sigmoid'))  # Output Layer ('NORMAL' or 'PNEUMONIA')
```

```
E:\anaconda3\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

In [12]:
```python
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

In [71]:
```python
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 238, 238, 32)	896
max_pooling2d (MaxPooling2D)	(None, 119, 119, 32)	0
conv2d_1 (Conv2D)	(None, 117, 117, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 58, 58, 64)	0
conv2d_2 (Conv2D)	(None, 56, 56, 128)	73,856
max_pooling2d_2 (MaxPooling2D)	(None, 28, 28, 128)	0
flatten (Flatten)	(None, 100352)	0
dense (Dense)	(None, 128)	12,845,184
dense_1 (Dense)	(None, 64)	8,256
dropout (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 1)	65

Total params: 38,840,261 (148.16 MB)

Trainable params: 12,946,753 (49.39 MB)

Non-trainable params: 0 (0.00 B)

Optimizer params: 25,893,508 (98.78 MB)

In [13]:
```python
from tensorflow.keras.callbacks import ModelCheckpoint
checkpoint = ModelCheckpoint(r'best_model.keras',
                             monitor='val_accuracy', save_best_only=True, mode='max', verbose=1)


history = model.fit(train_generator, validation_data=test_generator, epochs = 10, callbacks=[checkpoint],
                    shuffle = False)

# Load the best weights
model.load_weights('best_model.keras')
```

```
163/163 ━━━━━━━━━━━━━━━━━━━━ 0s 4s/step - accuracy: 0.8935 - loss: 0.2578
Epoch 6: val_accuracy did not improve from 0.87500
163/163 ━━━━━━━━━━━━━━━━━━━━ 742s 5s/step - accuracy: 0.8935 - loss: 0.2578 - val_accuracy: 0.8189 - val_loss: 0.3797
Epoch 7/10
163/163 ━━━━━━━━━━━━━━━━━━━━ 0s 5s/step - accuracy: 0.8862 - loss: 0.2742
Epoch 7: val_accuracy did not improve from 0.87500
163/163 ━━━━━━━━━━━━━━━━━━━━ 809s 5s/step - accuracy: 0.8863 - loss: 0.2740 - val_accuracy: 0.8734 - val_loss: 0.3192
Epoch 8/10
163/163 ━━━━━━━━━━━━━━━━━━━━ 0s 5s/step - accuracy: 0.9121 - loss: 0.2163
Epoch 8: val_accuracy improved from 0.87500 to 0.88141, saving model to best_model.keras
163/163 ━━━━━━━━━━━━━━━━━━━━ 786s 5s/step - accuracy: 0.9121 - loss: 0.2163 - val_accuracy: 0.8814 - val_loss: 0.3084
Epoch 9/10
163/163 ━━━━━━━━━━━━━━━━━━━━ 0s 6s/step - accuracy: 0.9086 - loss: 0.2203
Epoch 9: val_accuracy did not improve from 0.88141
163/163 ━━━━━━━━━━━━━━━━━━━━ 1028s 6s/step - accuracy: 0.9086 - loss: 0.2203 - val_accuracy: 0.8782 - val_loss: 0.3282
Epoch 10/10
163/163 ━━━━━━━━━━━━━━━━━━━━ 0s 5s/step - accuracy: 0.9177 - loss: 0.2044
Epoch 10: val_accuracy did not improve from 0.88141
163/163 ━━━━━━━━━━━━━━━━━━━━ 820s 5s/step - accuracy: 0.9177 - loss: 0.2044 - val_accuracy: 0.8365 - val_loss: 0.4421
```

Saving the model

In [14]:
```python
model.save('my_model.keras')
```

In [15]:
```python
new_model=keras.models.load_model('my_model.keras')
```

Model-evaluation

In [16]:
```python
new_model.evaluate(test_generator)
```

```
20/20 ━━━━━━━━━━━━━━━━━━━━ 34s 2s/step - accuracy: 0.8975 - loss: 0.2990
```

Out[16]: `[0.308383047580719, 0.8814102411270142]`

In [17]:
```python
new_model.evaluate(train_generator)
```

```
163/163 ━━━━━━━━━━━━━━━━━━━━ 487s 3s/step - accuracy: 0.9206 - loss: 0.1797
```

Out[17]: `[0.18747134506702423, 0.9177530407905579]`

In [18]:
```python
new_model.evaluate(val_generator)
```

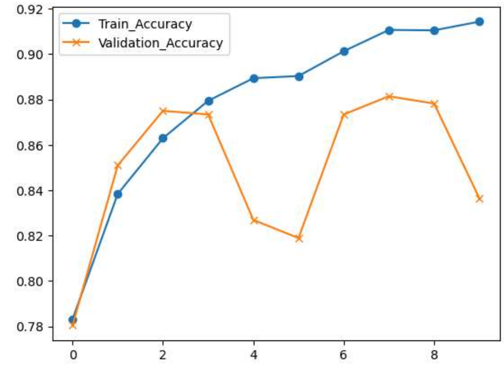
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 1s/step - accuracy: 0.5625 - loss: 0.8518
```

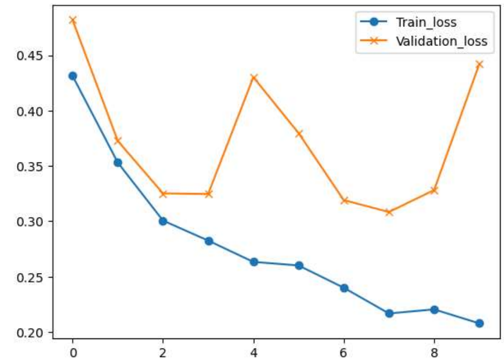
Out[18]: `[0.8517850041389465, 0.5625]`

In [22]:

```python
# Plot accuracy and loss
import matplotlib.pyplot as plt
plt.plot(history.history['accuracy'], marker ='o' ,label ='Train_Accuracy')
plt.plot(history.history['val_accuracy'], marker ='x',label ='Validation_Accuracy')
plt.legend()
plt.show()

# Similar plot for loss
plt.plot(history.history['loss'], marker ='o',label ='Train_loss')
plt.plot(history.history['val_loss'],marker ='x',label ='Validation_loss')
plt.legend()
plt.show()
```





Make Predictions

In [53]:

```python
import os
# Load and preprocess a single image
img_paths= [os.path.join(r'E:\DS journey\Deep Learning Datasets\chest_xray\train\PNEUMONIA', i) \
            for i in os.listdir(r'E:\DS journey\Deep Learning Datasets\chest_xray\train\PNEUMONIA')]
len(img_paths)
```

Out[53]: 3875

In [68]:

```python

img_path_=img_paths[2500:2550]  # enter indices here
for img_path in img_path_:

    # Load and preprocess a single image
    img = tf.keras.preprocessing.image.load_img(img_path, target_size=(img_height, img_width))
    img_array = tf.keras.preprocessing.image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0) / 255.0
    #plt.imshow(img_array.reshape(240,240,3))
    # Predict
    prediction = model.predict(img_array)
    print(prediction)
    print('NORMAL' if prediction <0.5 else 'PNEUMONIA')
```

```
PNEUMONIA
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 337ms/step
[[0.9806409]]
PNEUMONIA
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 457ms/step
[[0.9757314]]
PNEUMONIA
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 189ms/step
[[0.8217449]]
PNEUMONIA
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 373ms/step
[[0.9924891]]
PNEUMONIA
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 447ms/step
[[0.74353504]]
PNEUMONIA
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 293ms/step
[[0.9882326]]
PNEUMONIA
```

Observations:

1. This is an Image Classification Porblem
2. The CNN Model takes the Xray scan image as input and predicts if the patient is NORMAL or infected with PNEUMONIA
3. The Model performance is as foolows:
- trianing prediction accuracy of 92.06% with loss of 0.1797 and
- test preiction accuracy of 89.75% with loss of 0.2990
4. Accuracy can be further improved by tuning the model paramaters as:
- optimizer
- learning_rate
- no. of convolution and pooling layers
- no. of Dense layers